LA-UR-02-2288

How does ASCI actually complete multi-month 1000-processor milestone simulations?

Conference on High Speed Computing
April 22-25, 2002

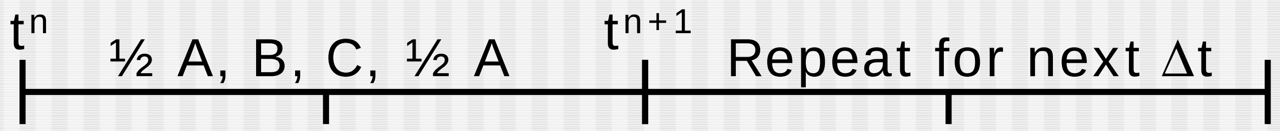
Ken Koch
X-DO ASCI Applications Program Manager
Los Alamos National Laboratory

Topics covered

1. general code issues
2. running Tera-scale milestone simulations
3. machine configuration and operational issues
4. some pitfalls

ASCI Simulation Codes 101

- Multiple Coupled Physics
  - Time-marching on spatial grids
    - 1D, 2D, 3D
    - Cartesian, Unstructured, Continuous-AMR
  - Physics A, B, C, … (local, explicit, implicit, regional)



- Programming Languages
  - F90 and/or C++ predominantly; some C for I/O etc.
  - mixed languages is typical
  - all data “containers” are dynamically allocated
  - most data accessed by indirection memory references

ASCI Simulation Codes 101

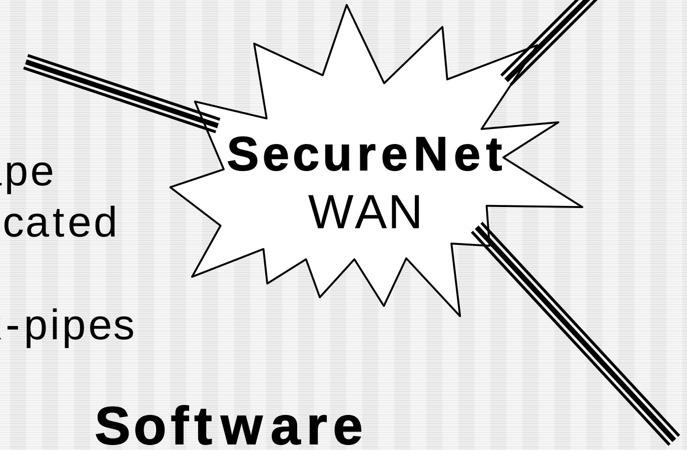
- MPI distributed memory parallelism
  - domain decomposition of spatial grids (predominately)
    - Some use of decomposition in other domains (e.g. energy & space, particles)
  - nearest-neighbor exchanges
    - “some-to-some” or gather/scatter via point-to-point send/recv communications lists
    - these exchanges essentially synchronize the overall program flow (SPMD-like)
  - Significant use of all_reduce for global control/monitor variables and in implicit solvers (e.g. CG scaling)
  - LLNL has used OpenMP with MPI

ASCI Simulation Codes 101

- Restart ("checkpoint") dump files
  - all nodes participate at given time step intervals
  - I/O models
    - local I/O per MPI process to individual files
    - data aggregation to single MPI process to single file
    - data aggregation to few MPI processes to single file via multiple I/O adaptors or multiple nodes
    - parallel HDF5 over MPI-IO (recent)
- Graphics dumps done similarly or typically a restart dump serves both purposes

ASCI Tri-Lab Environment

LANL
ASCI Q (Compaq)
  3 x 1024 x 4-way ES45
  w/ Quadrics
Blue Mountain (SGI)
  48 128-way O2K
  w/ HiPPI
NFS servers
HPSS archival disk/tape
2 BlueMtn boxes dedicated
  as viz servers
powerwalls, RAVE, IR-pipes
  video distribution



LLNL
ASCI White (IBM)
  512 x 16-way SP2
Blue Pacific (IBM)
  3 x 512 x 4-way SP2
NFS & DFS servers
HPSS archival disk/tape
O2K viz servers
powerwalls, IR-pipes video
  distribution

SNL
ASCI Red (Intel)
  ~4500 x 2-way MPP
NFS & DFS servers
HPSS archival disk/tape
O2K & cluster viz servers
powerwalls

Software
MPI,OpenMP,KCC,GCC
MPI-IO, parallel HDF5
LSF, DPCS
Totalview, Vampir
EnSight Gold, MeshTV

A node dies!

- Loss of a single node causes blockage of the overall simulation
  - data is lost and must be recovered or regenerated
  - key physics require neighbor exchanges or global reductions (if implicit)
  - some MPI requests can’t complete
- Domain decomposition spreads vital data across all nodes
  - each spatial cell exists in **one and only one** processor’s memory (except possible ghost or halo cells)

High Availability Approach

- What would be needed?
  - provide duplicate of all data quantities in memory
    - 2X memory required
    - impossible for one node to hold all data as a backup
    - “slave” duplicate data or develop new overlapping decomposition methods (double assignment)
  - dynamically recoverable and reconfigurable MPI
  - resync all processes to know condition
    - possibly rollback state of remaining N-1 processes (all variables & unwind their call trees)
  - possibly request extra node from resource manager in real time; otherwise redistribute data from N to N-1 domains
- Do this all in a verified manner for all physics modules with limited software developers

Restart Approach

- Use a checkpoint/restart capability!
  - let job die and resubmit a restart job
  - checkpoint/restarts are a normal way of doing business anyway
  - sometimes there are common-mode failures across many nodes
    - only waiting for system recovery helps
- Not elegant, but far easier with a proven track record (to-date)
- A minimum mean-time-to-interrupt of a few days (on 1/5 to 1/2 of system) is generally sufficient to prevent churning

Milestone Characteristics

- Few million to ½ billion cell 3D problems
- >500 to 4000 processors
- ½ to 2 GB per processor of data variables

- Weeks to months to complete just one simulation
- Machines and environment not fully mature

Milestone Characteristics

- Remote Classified machines
- Comparisons to experimental data

- Los Alamos, Livermore, and Sandia have each completed several and are signed up for one per year

LANL Milestone Example

- 3D “full-system” (primary + secondary) nuclear package explosion simulation (Dec01 Milestone)
  - all processes from initial detonation to full explosive yield
  - LANL’s Crestone Project completed this; LLNL did it too
- Ran on Livermore’s ASCI White remotely from Los Alamos
  - ¼ of ASCI White (128 nodes) allocated to this effort
  - Ran from March-June and August-October 2001; July was for memory upgrades; completed 2 months early
  - ~2000-4000 processors; 123 wallclock days; 750 processor years; 10’s TB of files

Restart, restart, restart!

- Each job starts where last left off (optimally)
  - O(100-200) restarted jobs overall!
  - Some jobs are deemed "inadequate" and must be repeated with different options/settings
  - "steering" can be done between jobs (e.g. Δt or AMR settings)
- Automated "smart" job script
  - submit follow-on "dependant" job first; then run code for hours on end (chained jobs)
  - Pre- & post-run actions from user command files
  - Manual human archival of dump files done later
    - Done out-of-phase of the actual job chains

Single Job's Characteristics

- 24hrs runtime for each job submission (typical)
  - Some runs were for 48 & 96 hrs each
  - Jobs run until their time limit runs out
  - 1000's of processors using virtually all their memory
  - Much longer jobs are not necessarily beneficial!
- Write a restart dump file(s) every ~70mins
  - A single time step can be as long as 10mins
  - Two forms of restart files
    - an alternating A/B overwritten dump file pair
    - permanent non-overwritten dump file series
  - 100-250GB per restart file; used IBM GPFS parallel I/O at 1-2GB/s rates
  - Fears of problems led to (overly?) aggressive writing of restarts

Humans keep it going

- Dedicated “monitor(s)” of running jobs
  - “tail –f” of logfile and “ls -l”
  - Long hours
    - almost 24x7 at times
    - on-call via pagers and cell phones
- Machine operators helped monitor jobs
  - keep jobs running continuously in queues
  - call people when needed
  - read logfile “indicators” of classified run for someone at Los Alamos at their residence at night

Operational Issues

- Target machine availability & reliability to users
  - if the machine isn’t available AND working right, users aren’t getting results
  - reboot time for full systems are becoming outrageous
    - White & Q(1/6 scale) now plan for 4 hours!
  - little if any cluster-wide testing of parallel capability is “built in”
    - reliability is sometimes an afterthought
- Must support mixed workload
  - login, edit, compile, serial tests, serial production
  - small scale parallel production (10’s-256 processors)
  - large tera-scale testing and demonstrations (1000’s or processors
  - large scale (100’s processor) debugging

Operational Issues

- dedicated test periods for developers
  - regularly available by request of large dedicated machine partitions (50%–90% of whole machine)
  - code, system, & vendor staff work as a SWAT-team
- weekly preventative maintenance
  - two separate periods for hardware and software
  - special test-suite was developed at LANL to verify machine functionality

Pitfall #1 - failed job starts

- Parallel jobs fail soon after launch
  - large parallel job starts but then quickly fails
  - obtuse error message or none at all
  - continued identical resubmits may eventually get one to run properly!

- Parallel jobs fail to launch at all
  - processors are available but vendor & layered 3rd party queuing systems fail to start new jobs
  - nodes “die” and vendor & layered queuing systems get confused

Pitfall #1 - failed job starts

- Root Cause
  - inadequate testing of layered queuing systems at tera-scale configurations
  - system services “blink out” on some nodes
  - lack of meaningful error messages
  - lack of cluster-wide admin tools to maintain consistency
- Mitigation
  - humans must watch & “nudge” job launches
  - smart job scripts and automatic retries
  - retry failures, some of which turn out to be true bugs somewhere

Pitfall #2 - bad CPUs

- Bad CPU(s) in node(s)
  - Milestone run generates NANs in middle of run in non-repeatable fashion!
  - Code team reruns from multiple restarts using multiple executables
  - Cross-correlation points to suspect nodes which are removed from service
    - Milestone runs can continue without errors on new nodes
  - Suspect nodes
    - passed LINPACK and other applications code tests
    - must be tested in kernel-mode with vendor diagnostics - they pass!
    - “fixed” by vendor and returned to service

Pitfall #2 - bad CPUs

- Root cause
  - bad HW instances
  - per node probabilities do cause problems at large node count
  - adequate testing can’t be done in user-mode
  - no regularly scheduled functionality (verification) tests
- Mitigation
  - look for NaNs in code results
  - hope preventative maintenance is good enough
  - need reliable nodes

Pitfall #3 - bad interconnect

- Bad optical HiPPI terminations
  - led to “non-repeatable” HW link errors
  - firmware and MPI SW did not detect & abort
  - bad MPI data caused erroneous code results
  - immature & complex OO code base presumed at fault
  - code team spends days/weeks tracking down fault
  - code team writes own data integrity checksums

Pitfall #3 - bad interconnect

- Root Cause
  - MPI & interconnect HW design didn’t address this case
- Mitigation
  - MPI & low-level communications test code written and run regularly across entire machine
  - MPI design should allow data integrity check option

Pitfall #4 - bad I/O

- File system unavailable but program continues to run
  - Global parallel file-system drops out
  - Program runs through restart dump I/O calls without blocking or generating a system error!
  - No files produced
  - Code developer gets involved and helps test error actions associated with global parallel file system I/O

Pitfall #4 - bad I/O

- Root Cause:
  - system runtime I/O library design suspect
  - inadequate system integration testing
  - code doesn’t check for I/O errors directly
    - BUT testing showed that would not have mattered as no error condition was generated!
- Mitigation
  - backup to last restart and start over
  - bug fix I/O runtime for this issue(?)
  - presumed never to return

Pitfall #5 - bad archival

- HPSS archived restart files corrupted
  - code will not properly restart from some files retrieved from archival HPSS storage
  - code **does** restart from same file still residing on local scratch disks
  - testing various HPSS restart files points to multiple bad files
  - file compares prove corruption
  - Milestone runs continue without relying on HPSS; disks get pretty full
  - HPSS internal testing uncovers firmware bug on HPSS disks or controllers and estimated dates of vulnerability

Pitfall #5 - bad dump files

- Root Cause:
  - firmware bug in “support” (HPSS) system
  - no regularly scheduled functionality testing
- Mitigation
  - have to live with expected gap in simulation restart sequence
  - “once in a lifetime” bug?
  - Data integrity is key user expectation!

Key Issues to Remember

- Codes need to run in 1-3 day chunks with restarts in between
  - Human decisions are necessary
  - Jobs run to completion on dedicated nodes
- A single job is not mission critical and does not have to be high availability by itself
  - Human monitoring and control are necessary to keep chained jobs going

Key Issues to Remember

- Node drop outs require the simulation to stop
  - domain decomposition doesn’t result in redundant data
  - lost data variables can’t be ignored or regenerated
  - easiest to back up to last restart file dump time in a new job
  - requires modest several day RELIABLE system for the parts of the machine in use
- Machine resources are fully committed
  - no extra memory; no idle nodes
  - large memory sets and processor counts

(My) Perceived Weak Areas

- Job launching
  - better queuing system integration testing (and design?)
  - better error messages
  - cluster-wide admin tools are needed
- Inadequate “verification” testing of the machine environment
  - End users see errors that possibly could have shown up in testing
  - Applications code developers get drafted into helping identify problems
  - Machine test periods have been valuable to users
    - resolves problems quickly and more efficiently
    - process benefits all users

NNSA

Los Alamos
NATIONAL LABORATORY